

Description

Using Mobility Tokens to Observe Malicious Mobile Code

5 Inventor: Ahmed Sallam

Technical Field

This invention pertains generally to computer
10 security, and more specifically to using mobility tokens to
observe malicious mobile code.

Background Art

Networked computers are vulnerable to malicious
computer code attacks, such as worms, viruses and Trojan
15 horses. As used herein, "malicious computer code" is any
code that enters a computer without an authorized user's
knowledge and/or without an authorized user's consent.

Malicious mobile executable files comprise a major
security threat. Mobile executable files are typically not
20 signed, and do not carry a digital signature. Thus, the
identity of their author(s) is unknown, and they should be
suspected of being potentially malicious. For example,
worms are often spread as a form of malicious mobile
executable file.

Malicious mobile executable files such as worms often utilize open shares to spread themselves within networks.

However, legitimate code can also be copied within a network this way. Being able to track mobile executable

5 files within networks is advantageous in determining whether the code is malicious or not, and blocking and eliminating detected malicious mobile code.

To determine whether a mobile file is malicious, and how to eliminate threats from detected malicious mobile
10 files, it would be helpful to know the answers to questions such as:

- What is the identity of the user who first introduced the file into the network?
- From which computer did the file originate?
- 15 • When did the file originate?
- On to which other computers has the file been copied?
- What route has the file taken, over what time period?
- Is there a known malicious code signature associated with the file?
- 20 • Is that signature available on any of the computers on which the file is installed?

There is no mechanism within Microsoft Windows® or other operating systems to automatically determine or track

such information as a mobile file moves around a network. What is needed are methods, systems and computer readable media for tracking mobile files in real-time as they move throughout a network via open shares.

5

Disclosure of Invention

The present invention comprises methods, systems and computer readable media for tracking movement of files (105) within a network. A mobility token manager (101) on a source computer (113) detects an attempt to write a file (105) to a target computer (117). Responsive to the
10 detection, the mobility token manager (101) writes a mobility token (103) containing data concerning at least the file (105) and the write operation to the target computer (117).

15 A mobility token manager (101) on the target computer (117) detects that the mobility token (103) is being written to the target computer (117). The mobility token manager (101) on the target computer (117) reads the mobility token (103), and determines relevant information
20 concerning the file (105) associated with the mobility token (103).

In some embodiments, mobility token managers (101) are instantiated within a file system filter driver (107). In

other embodiments, mobility token managers (101) are instantiated as at least one system call wrapper (301).

The features and advantages described in this disclosure and in the following detailed description are not all-inclusive, and particularly, many additional features and advantages will be apparent to one of ordinary skill in the relevant art in view of the drawings, specification, and claims hereof. Moreover, it should be noted that the language used in the specification has been principally selected for readability and instructional purposes, and may not have been selected to delineate or circumscribe the inventive subject matter, resort to the claims being necessary to determine such inventive subject matter.

Brief Description of the Drawings

Figure 1 is a block diagram illustrating a high level overview of a source computer processing mobility tokens, according to some embodiments of the present invention.

Figure 2 is a block diagram illustrating a high level overview of a target computer processing mobility tokens, according to some embodiments of the present invention.

Figure 3 is a block diagram illustrating an embodiment of the present invention in which a mobility token manager is implemented in the form of system call wrappers.

The Figures depict embodiments of the present invention for purposes of illustration only. One skilled in the art will readily recognize from the following discussion that alternative embodiments of the structures and methods illustrated herein may be employed without departing from the principles of the invention described herein.

Detailed Description of the Preferred Embodiments

Figure 1 illustrates a high level overview of a system 100 for performing some embodiments of the present invention. A mobility token manager 101 uses mobility tokens 103 to track mobile files 105. It is to be understood that although the mobility token manager 101 is illustrated as a single entity, as the term is used herein a mobility token manager 101 refers to a collection of functionalities which can be implemented as software, hardware, firmware or any combination of the three. Where a mobility token manager 101 is implemented as software, it can be implemented as a standalone program, but can also be implemented in other ways, for example as part of a larger

program, as a plurality of separate programs, or as one or more statically or dynamically linked libraries.

In Figure 1, the mobility token manager 101 is illustrated as being within a file system filter driver 107. The implementation mechanics of file system filter drivers 107 are known to those of ordinary skill in the relevant art. For example, in Microsoft Windows® NT and related versions of Microsoft Windows®, a network file system filter driver 107 can be attached to the Microsoft LANMAN network redirector (mrxsmb.sys). In other embodiments, the mobility token manager 101 is not implemented within a file system filter driver 107, but instead in other ways, for example as a collection of system call wrappers, as discussed in greater detail below.

As illustrated in Figure 1, the mobility token manager 101 uses mobility tokens 103 (special purpose files described in greater detail below) to track mobile files 105 (e.g., files being moved within a network). Consider the scenario illustrated in Figure 1, wherein an application program 109 running in user space 111 on a source computer 113 makes a write request 115 to copy a file 105 to a remote target computer 117 (not illustrated in Figure 1) via an open share. Such a request 115 could have the form, for example, of "copy file_name

\\target_computer_name_\\open_share_name\\local_file_name."

The request 115 is then processed by the operating system input output (I/O) manager 119 running in kernel space 121, which in turn passes it to the file system driver 123 (the
5 file system driver 123 utilized in the case of copying to a remote computer 117 through an open share under Windows® would be the Windows file system driver network redirector mrxsmbs.sys). The I/O manager 119 illustrated in Figure 1 would be present in various versions of Windows®, but other
10 embodiments of the present invention are implemented under operating systems which may not include the illustrated I/O manager 119. As illustrated, the mobility token manager
101 on the source computer 113 is implemented as a part of a file system filter driver 107, which is attached to the
15 file system driver 123, and filters all of the file system driver's 123 throughput.

Because mobile files 105 being written to a target computer 117 go through the file system driver 123 of the source computer 113, the mobility token manager 101
20 installed in the filter driver 107 attached thereto can recognize all such mobile files 105, at write time.

When the mobility token manager 101 recognizes a mobile file 105 being written to a target computer 117 through the file system driver 123, the mobility token

manager 101 writes another file (a mobility token 103) to the target computer 117 to which the mobile file 105 is written. Since the filter driver 107 of which the mobility token manager 101 is a part is already writing to the
5 target computer 117, then it is guaranteed that it has proper write access to the target folder of this remote location. In various embodiments, the mobility token manager 101 writes mobility tokens 101 concerning some or all mobile files 105 being written to target computers 117
10 as desired.

The mobility token 103 can contain information concerning the source computer 113 (e.g., IP address, computer name, primary domain controller name, etc.), information concerning the written mobile file 105 (e.g.,
15 file name, content-based hash value, digital signature information, version information, last modification date, last modification time, etc.), information concerning the user who has ownership of the application program 115 that requested the write operation (e.g., user account name,
20 user account number, SID, etc.) plus any other additional relevant information as desired (e.g., date and time of the write operation, etc.). Of course, in various embodiments mobility tokens 103 contain more or less information as desired. Because each user typically has a byte quota that

it can write to the target machine, mobility tokens 103 are kept small. The specific size limitation is a variable design choice. Mobility tokens 103 can but need not be compressed in order to save space. Additionally, mobility
5 tokens 103 typically will be secured through standard cryptography techniques.

As illustrated in Figure 2, in some embodiments a mobility token manager 101 will also be instantiated on the target computer 117. In some embodiments, the mobility
10 token manager 101 on a source computer 113 only generates a mobility token 103 when there is another mobility token manager 101 running on the target computer 117.

Upon the close or cleanup of any remote file operation, the mobility token manager 101 on the source
15 computer 113 can close the mobility token 103 on the target computer 117 that corresponds to the written mobile file 105, before or as the file close or cleanup operation is committed. Since the mobility token manager 101 on the source computer 113 holds an exclusive lock to be able to
20 write to the target mobile file 105, then it can be assumed that it can write the associated mobility token 103 as well.

The mobility token manager 101 running on the target computer 117 recognizes that a mobility token 103 is being

written locally. The mobility token manager 101 on the target computer 113 can recognize the mobility token 103 by name, content based signature, size, etc. By reading the information stored in the mobility token 103, the mobility token manager 101 can determine relevant information concerning the recently copied mobile file 105. The mobility token manager 101 on the target computer 117 can read the mobility token 101, for example, when the associated mobile file 105 is closed, or even before it is closed, depending on the desired security policy.

In one embodiment, a new entry is generated and written to a mobility token 103 for every file "open-write-close" session concerning its associated file 105. In some embodiments, there is a maximum number of entries per mobility token 103, so as to keep the token 103 size small. In such embodiments, typically older entries are overwritten when the maximum number of entries is reached. Of course, in other embodiments other cache management techniques can be used as desired.

In some embodiments, the mobility token manager 101 on the target computer 117 merges data from multiple mobility tokens 101, and stores the data in a separate mobility token data store 201 (e.g., a database). In some embodiments, the mobility token manager 101 maintains a

mobility token 103 for at least some folders in the file system, as well as for individual files.

The mobility token manager 101 on the target computer 117 can optionally hide mobility tokens 103 so that they cannot be seen or read by any process other than a mobility token manager 101. Implementing file hiding by a file system filter driver 107 is known to those of ordinary skill in the relevant art. In such embodiments, requests to list file names within any folder or directory will not return the name of any hidden mobility token 103, and requests to access a hidden mobility token 103 by name will generate an error condition. In some embodiments, all, some or no mobility tokens 103 are hidden as desired.

When a mobile file 105 is copied from a source computer 113 to a target computer 117, the mobility token manager 101 on the source computer 113 checks to see whether it has an existent associated mobility token 103 for the file 105 to be copied. If an associated token 103 exists, then the mobility token manager 101 will add current information (e.g., information concerning the file 105 and the current write operation) to the mobility token 103. If no mobility token 103 exists for the mobile file 105, then the mobility token manager 101 on the source computer will create a new mobility token 103 concerning

the file 105, and write that new mobility token 103 to the target computer 117 with the mobile file 105.

Additionally, the mobility token manager 101 on the source computer 113 can write instructions in a mobility token 103 to instruct the mobility token manager 101 on the target computer 117 to perform or not to perform desired actions. For example, the mobility token manager 101 on the source computer 113 could instruct the mobility token manager 101 on the target computer 117 not to copy the associated mobile file 105 to any additional computers, or to copy it to another specific computer, or to send an acknowledgement of the receipt of the file to a third computer. This type of instructive control can be at any level of detail as desired. For example, a mobility token 103 could include instructions preventing certain source computers 113 from copying an associated file 105 to specific folders on specific target computers 117. Time ranges can also be blacked out or approved for specific movements of files 105 within a network. Of course, the above listed directed actions are examples, and other actions are possible and within the scope of the present invention.

Mobility token managers 101 on target computers 117 can also control incoming network file 105 traffic by

reading mobility tokens 103. For example, a target computer 117 could reject files 105 from non-trusted computers, or that have been touched non-trusted users. These are again simply examples of the actions a mobility token manager 101 on a target computer 117 can take responsive to the contents of an incoming mobility token 103.

In one embodiment, at least some mobility tokens 103 contain an indication of whether their associated file 105 has been scanned by an anti-virus scanning engine, and if so an identifier of the virus definition file that was used. This information can be used to scan files 105 as necessary, and to save time and resources by not scanning files 105 unnecessarily as they are moved about within a network.

In some embodiments, the mobility token manager 101 is instantiated in the form of system call wrappers 301, as illustrated in Figure 3. The implementation mechanics of intercepting system calls 303 by overwriting pointers 305 in an interrupt vector table 307 and redirecting the pointers 305 to system call wrapper 301 is known to those of ordinary skill in the relevant art. In such embodiments, system calls 303 to write and read files 105 are intercepted as desired, and one or more system call

wrappers 301 execute the above describe functionality of a mobility token manager 101. Because system calls 303 that read and write files 105 are intercepted, attempts to transfer mobile files 105 through open shares are detected, and corresponding mobility tokens 103 are processed as described above.

As will be understood by those familiar with the art, the invention may be embodied in other specific forms without departing from the spirit or essential characteristics thereof. Likewise, the particular naming and division of the modules, managers, features, attributes, methodologies and other aspects are not mandatory or significant, and the mechanisms that implement the invention or its features may have different names, divisions and/or formats. Furthermore, as will be apparent to one of ordinary skill in the relevant art, the modules, managers, features, attributes, methodologies and other aspects of the invention can be implemented as software, hardware, firmware or any combination of the three. Of course, wherever a component of the present invention is implemented as software, the component can be implemented as a script, as a standalone program, as part of a larger program, as a plurality of separate scripts and/or programs, as a statically or dynamically linked library, as

a kernel loadable module, as a device driver, and/or in every and any other way known now or in the future to those of skill in the art of computer programming. Additionally, the present invention is in no way limited to

5 implementation in any specific programming language, or for any specific operating system or environment. Accordingly, the disclosure of the present invention is intended to be illustrative, but not limiting, of the scope of the invention, which is set forth in the following claims.

10 What is claimed is: